Decision Trees for Mining Data Streams Based on the McDiarmid's Bound

Leszek Rutkowski, *Fellow*, *IEEE*, Lena Pietruczuk, Piotr Duda, and Maciej Jaworski

Abstract—In mining data streams the most popular tool is the Hoeffding tree algorithm. It uses the Hoeffding's bound to determine the smallest number of examples needed at a node to select a splitting attribute. In the literature the same Hoeffding's bound was used for any evaluation function (heuristic measure), e.g., information gain or Gini index. In this paper, it is shown that the Hoeffding's inequality is not appropriate to solve the underlying problem. We prove two theorems presenting the McDiarmid's bound for both the information gain, used in ID3 algorithm, and for Gini index, used in Classification and Regression Trees (CART) algorithm. The results of the paper guarantee that a decision tree learning system, applied to data streams and based on the McDiarmid's bound, has the property that its output is nearly identical to that of a conventional learner. The results of the paper have a great impact on the state of the art of mining data streams and various developed so far methods and algorithms should be reconsidered.

Index Terms—Data streams, decision trees, Hoeffding's bound, McDiarmid's bound, information gain, Gini index

1 Introduction

1.1 Motivation and Results

Decision trees developed in the 80s and the 90s, e.g., ID3, C4.5, and Classification and Regression Trees (CART), are powerful techniques in data mining. At the beginning of 2000s, they have been adapted to deal with stream data [1], [2], [4], [8], [9]. The problem was to ensure that, with high probability, the attribute chosen using N examples is the same as that chosen using infinite examples. The goal was to compute the heuristics measures, e.g., information gain or Gini index, based on these N examples, and then to split the examples (learning sequence) according to this attribute. To solve the problem hundreds of researches used the so-called "Hoeffdings trees," derived from the Hoeffding's bound, for mining data streams. The Hoeffding's bound states that with probability $1-\delta$ the true mean of the random variable of range R does not differ from the estimated mean, after N independent observations, by more than:

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}. \tag{1}$$

A close look at the techniques and reasoning of data stream researches motivated us to revise their methodology. In this paper, we show that:

1. The Hoeffding's inequality is not an adequate tool to solve the underlying problem and the previously obtained and presented in the literature results, based on it, in a general case (for any heuristic measure) are not true. Consequently, all methods and algorithms developed in the literature for mining data streams, if they use the Hoeffding's bound, should be revised.
2. The McDiarmid's inequality, used in a proper way, is an effective tool for solving the problem.
3. The well-known result "Hoeffding's bound," with ϵ given by (1), for an arbitrary heuristic measure, cannot be obtained either from the Hoeffding's inequality (see Section 1.3) or from the McDiarmid's inequality (see Section 2).
4. The McDiarmid's bound for information gain (used in ID3 algorithm) is given by (29) and (30), see Theorem 1 in Section 3. It is easily seen that our result differs from (1).
5. The McDiarmid's bound for Gini index (used in CART algorithm) is given by (37) and (38), see Theorem 2 in Section 4. Formulas (37) and (1) are, by chance, of the same form. However, it should be emphasized that (1) cannot be derived from the Hoeffding's inequality, as it was argued by previous authors.

Without any doubts, the results of the paper have a great impact on the state of the art of mining data streams and it shows that various developed so far methods and algorithms should be reconsidered.

1.2 Decision Trees for Nonstream Data

Data mining is an interdisciplinary branch of science drawing ideas from various methods, areas and techniques such as regression analysis and generally statistics, machine learning, database systems, pattern recognition, signal and image processing and soft computing [10], [22], [23], [15]. Typical applications include prediction of sales in a supermarket or clustering bank's clients into groups of a risk. One

• *L. Rutkowski is with the Institute of Computational Intelligence, Czestochowa University of Technology, ul. Armii Krajowej 36, 42-200 Czestochowa, Poland, and Information Technology Institute, Academy of Management, 90-113 Łódź, Poland.*
E-mail: leszek.rutkowski@iisi.pcz.pl.

• *L. Pietruczuk, P. Duda, and M. Jaworski are with the Institute of Computational Intelligence, Czestochowa University of Technology, ul. Armii Krajowej 36, 42-200 Czestochowa, Poland.*
E-mail: {lena.pietruczuk, piotr.duda, maciej.jaworski}@iisi.pcz.pl.

Manuscript received 14 July 2011; revised 26 Oct. 2011; accepted 24 Jan. 2012; published online 9 Mar. 2012.
Recommended for acceptance by S. Greco.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-07-0410.
Digital Object Identifier no. 10.1109/TKDE.2012.66.

of the most popular techniques in data mining is based on a decision tree induction. Historically, the first decision tree algorithm was ID3 (Iterative Dichotomiser) [20], later C4.5 [21] and Classification and Regression Trees [5] algorithms were developed. In the beginning they were applied to solve problems characterized by static data. In most algorithms, e.g., ID3, entropy was taken as a measure of the impurity in a collection of training examples. As a measure of the effectiveness of an attribute in classifying the training data, the so-called information gain was used. Its interpretation is the expected reduction in entropy caused by splitting the examples according to this attribute. We will now formally describe this method and introduce the notation used through the rest of the paper.

Suppose that attribute a can take one of $|a|$ different values from the set $A = \{a_1, \dots, a_{|a|}\}$ (analogously for any other attribute) and $\Lambda = \{k_1, \dots, k_K\}$ is a set of different classes. Let

$$Z = \{X_1, \dots, X_i, \dots, X_N\}, \tag{2}$$

be the training set (set of examples) of size N, where $X_1, \dots, X_N$ are independent random variables defined as follows:

$$X_i = (a_{j_i}, b_{l_i}, \dots, k_{q_i}) \in A \times B \times \cdots \times \Lambda, \tag{3}$$

for $i = 1, \dots, N$, $j_i \in \{1, \dots, |a|\}$, $l_i \in \{1, \dots, |b|\}, \dots, q_i \in \{1, \dots, K\}$.

Each element of Z belongs to one of the K different classes k_j. Entropy associated with the classification of Z is defined as

$$H(Z) = -\sum_{j=1}^{K} p_j \log_2 p_j, \tag{4}$$

where p_j is the probability that element from Z comes from class k_j. We estimate this probability by $\frac{n^j}{N}$, where n^j is the number of elements from class k_j. Then

$$H(Z) = -\sum_{j=1}^{K} \frac{n^j}{N} \log_2 \frac{n^j}{N}. \tag{5}$$

Let us choose an attribute a, characterizing the elements of set Z. Then Z_{a_i} denotes a set of elements from Z, for which the value of a is a_i. The number of elements from set Z_{a_i} is labeled as n_{a_i}. Then the weighted entropy for attribute a and set Z is given by

$$H_a(Z) = \sum_{i=1}^{|a|} \frac{n_{a_i}}{N} H(Z_{a_i}), \tag{6}$$

where

$$H(Z_{a_i}) = -\sum_{j=1}^{K} \frac{n^j_{a_i}}{n_{a_i}} \log_2 \frac{n^j_{a_i}}{n_{a_i}}, \tag{7}$$

and $n^j_{a_i}$ denotes the number of elements in set Z_{a_i} from class k_j. Information gain for attribute a is given by

$$Gain_a(Z) = H(Z) - H_a(Z). \tag{8}$$

Let us assume, that a is an attribute with the highest value of information gain, while b is the second-best attribute. Define

$$\begin{aligned} f(Z) &= Gain_a(Z) - Gain_b(Z) \\ &= H(Z) - H_a(Z) - (H(Z) - H_b(Z)) \\ &= H_b(Z) - H_a(Z). \end{aligned} \tag{9}$$

In Section 4, we will extend the above notation to deal with another heuristic measure—Gini index.

1.3 Decision Trees for Stream Data and Hoeffding's Inequality

Traditional techniques for data mining require multiple scans of data to extract the information, which is not feasible for stream data. It is not possible to store an entire data stream or to scan through it multiple times due to its tremendous volume. The amount of events in data streams which had previously happened, is usually extremely large. Moreover, the characteristics of the data stream can vary over time and the evolving pattern needs to be captured [7], [14], [16], [24], [25], [26], [27]. To address these problems, as we indicated, several methods, including Hoeffding's tree algorithm, Very Fast Decision Tree (VFDT) and Concept-adapting Very Fast Decision Tree (CVFDT), have been developed and studied [6], [12], [13], [19], [17], [3]. These methods were supported mathematically by the Hoeffding's inequality [11]. Unfortunately, the authors applying the Hoeffding's inequality did not carefully check whether this probabilistic model is adequate to the descriptions of ID3 or C4.5, or CART algorithms.

Let $Y_1, Y_2, \dots, Y_N$ be random variables with real values from a certain distribution. Let $Y_i \in [0, R]$ for $i = 1, \dots, N$ and some constant R. Let us denote the expected value of this distribution by $E[Y]$ and the sample mean by $\overline{Y}$. The sample mean differs from the expected value by at least $\epsilon \geq 0$, with probability less than δ, i.e.

$$Pr(\overline{Y} - E[Y] \geq \epsilon) \leq \delta. \tag{10}$$

Since $\overline{Y} = \frac{\sum_{i=1}^{N} Y_i}{N}$, we can rewrite (10) as follows:

$$Pr(\overline{Y} - E[Y] \geq \epsilon) = Pr\left(\frac{\sum_{i=1}^{N} Y_i}{N} - E[Y] \geq \epsilon\right) \leq \delta. \tag{11}$$

Formula (11) is equivalent to

$$Pr\left(\sum_{i=1}^{N} Y_i - N \cdot E[Y] \geq N\epsilon\right) \leq \delta. \tag{12}$$

We will now recall the Hoeffding's inequality.

Hoeffding's inequality

Let Y_i for $i = 1, \dots, N$ be independent random variables such that $Pr(Y_i \in [a_i, b_i]) = 1$. Then for $S = \sum_{i=1}^{N} Y_i$ for all $\epsilon > 0$ we have the inequalities

$$Pr(S - E[S] \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right), \tag{13}$$

$$Pr(|S - E[S]| \geq \epsilon) \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right). \tag{14}$$

Obviously Hoeffding's inequality holds only for real valued data.

If each Y_i comes from the same distribution, then $E[Y] = E[Y_1] = \cdots = E[Y_N]$. Hence, $E[S] = N \cdot E[Y]$. Applying Hoeffding's inequality (13) to (12) we get

$$\begin{aligned} Pr(S - E[S] \geq N\epsilon) &\leq \exp\left(\frac{-2N^2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right) \\ &\leq \exp\left(\frac{-2N^2\epsilon^2}{\sum_{i=1}^{N} R^2}\right). \end{aligned} \tag{15}$$

Simplifying the denominator and equating the right side of (15) to δ we obtain

$$\exp\left(\frac{-2N\epsilon^2}{R^2}\right) = \delta. \tag{16}$$

Solving (16) with respect to ϵ, one gets (1).

Therefore, when N tends to infinity, ϵ tends to 0. Unfortunately, by analyzing descriptions (2)-(9) it can be easily seen that they do not fit the Hoeffding's inequality probabilistic model. First, only numerical data are applicable to the Hoeffding's inequality. In the general case the data do not have to be numerical. Second, split measures, like information gain and Gini index, cannot be expressed as a sum S of elements Y_i. Moreover they are using only the frequency of elements. The solution for this problem seems to be an application of the McDiarmid's inequality instead of the Hoeffding's inequality.

The rest of the paper is organized as follows: in Section 2, the McDiarmid's inequality is quoted. Sections 3 and 4 present the McDiarmid's bound for information gain and Gini index, respectively. In Section 5, following the idea of Domingos and Hulten [6], the McDiarmid Decision Tree algorithm is described. Experimental results are given in Section 6. The conclusions are drawn in Section 7.

2 McDiarmid's Inequality

Let Z, given by (2), be the set of independent random variables, with X_i taking values in a set A_i for each i. Let us define

$$Z' = \{X_1, \ldots, \hat{X}_i, \ldots, X_N\}, \tag{17}$$

with $\hat{X}_i$ taking values in A_i. Observe that Z' differs from Z, given by (2), only in the ith element.

In this paper, the basic tool to analyze data streams is the following McDiarmid's inequality [18].

McDiarmid's inequality

Suppose that the (measurable) function $\tilde{f}\colon \prod A_i \to \mathbb{R}$ satisfies

$$\sup_{X_1,\ldots,X_N,\widehat{X_i}} |\tilde{f}(Z) - \tilde{f}(Z')| \leq c_i, \tag{18}$$

for some constant c_i. Then

$$\begin{aligned} &Pr(\tilde{f}(Z) - E[\tilde{f}(Z)] \geq \epsilon) \\ &\quad \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N} c_i^2}\right). \end{aligned} \tag{19}$$

Remark. The Hoeffding's inequality (13) is a special case of McDiarmid's inequality (19) when $X_i \in A_i = [a_i, b_i]$, for $i = 1, \ldots, N$, are real valued random variables and $\tilde{f}(Z) = \sum_{X_i \in Z} X_i$.

Let us assume that $\tilde{f}$ is the function f defined by (9). Since $f(X_1, \ldots, X_N)$ is a function of random variables, it is a random variable as well. Then $E[f(X_1, \ldots, X_N)]$ is its expected value. To obtain the same form of ϵ as in (1), the following condition should be satisfied:

$$\sup_{X_1,\ldots,X_N,\widehat{X_i}} |f(Z) - f(Z')| \leq \frac{C}{N}, \tag{20}$$

for some constant C. Then,

$$\begin{aligned} &Pr(f(Z) - E[f(Z)] \geq \epsilon) \\ &\quad \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N} \frac{C^2}{N^2}}\right) = \delta, \end{aligned} \tag{21}$$

and we can get ϵ given by (1), if $C = R$.

Next it will be shown that assumption (20) does not hold and result (1) cannot be derived using the McDiarmid's inequality.

Let us assume that each element X from set Z comes from class k_1 and its value of attribute a is equal to a_1. The value of attribute b for all elements is equal to b_1, apart from element X_p, for which it is b_2, i.e.,

$$X_i = \begin{cases} (a_1, b_1, k_1), & \text{for } i \in \{1, \ldots, N\} \setminus \{p\}, \\ (a_1, b_2, k_1), & \text{for } i = p. \end{cases} \tag{22}$$

Then, in view of (6), (7), and (9) we have

$$\begin{aligned} f(Z) &= Gain_a(Z) - Gain_b(Z) \\ &= H_b(Z) - H_a(Z) \\ &= -\sum_{i=1}^{|b|} \frac{n_{b_i}}{N} \sum_{j=1}^{K} \frac{n_{b_i}^j}{n_{b_i}} \log_2 \frac{n_{b_i}^j}{n_{b_i}} \\ &\quad + \sum_{i=1}^{|a|} \frac{n_{a_i}}{N} \sum_{j=1}^{K} \frac{n_{a_i}^j}{n_{a_i}} \log_2 \frac{n_{a_i}^j}{n_{a_i}} \\ &= -\frac{1}{N}\frac{1}{1}\underbrace{\log_2 \frac{1}{1}}_{0} - \frac{N-1}{N}\frac{N-1}{N-1}\underbrace{\log_2 \frac{N-1}{N-1}}_{0} \\ &\quad + \frac{N}{N}\frac{N}{N}\underbrace{\log_2 \frac{N}{N}}_{0} = 0. \end{aligned} \tag{23}$$

Next, the element $X_p = (a_1, b_2, k_1)$ is replaced by $\hat{X}_p = (a_1, b_2, k_2)$. The value of f for set Z' is given by

$$\begin{aligned} f(Z') &= Gain_a(Z') - Gain_b(Z') \\ &= H_b(Z') - H_a(Z') \\ &= -\sum_{i=1}^{|b|} \frac{n_{b_i}}{N} \sum_{j=1}^{K} \frac{n_{b_i}^j}{n_{b_i}} \log_2 \frac{n_{b_i}^j}{n_{b_i}} \\ &\quad + \sum_{i=1}^{|a|} \frac{n_{a_i}}{N} \sum_{j=1}^{K} \frac{n_{a_i}^j}{n_{a_i}} \log_2 \frac{n_{a_i}^j}{n_{a_i}} \end{aligned}$$

$$\begin{aligned} &= -\frac{1}{N}\frac{1}{1}\underbrace{\log_2\frac{1}{1}}_{0} - \frac{N-1}{N}\frac{N-1}{N-1}\underbrace{\log_2\frac{N-1}{N-1}}_{0} \\ &\quad + \frac{N}{N}\left[\frac{N-1}{N}\log_2\frac{N-1}{N} + \frac{1}{N}\log_2\frac{1}{N}\right] \\ &= \frac{1}{N}\left[(N-1)\log_2\frac{N-1}{N} + \log_2\frac{1}{N}\right]. \end{aligned} \tag{24}$$

From (23) and (24) we get

$$|f(Z) - f(Z')| = \left|\frac{N-1}{N}\log_2\frac{N}{N-1} + \frac{\log_2 N}{N}\right|. \tag{25}$$

It is easy to prove that

$$\log_2\left(\frac{n+1}{n}\right) \le \frac{\log_2 e}{n}. \tag{26}$$

In view of (26), for $N \ge 2$, we can bound (25) as follows:

$$\begin{aligned} \frac{1}{N} &\le \frac{\log_2 N}{N} \\ &\le \underbrace{\left|\frac{N-1}{N}\log_2\frac{N}{N-1} + \frac{\log_2 N}{N}\right|}_{|f(Z)-f(Z')|} \\ &\le \frac{\log_2 e}{N} + \frac{\log_2(N)}{N} \le \frac{3\log_2(N)}{N}. \end{aligned} \tag{27}$$

In this particular example it is shown that the bound given in the form C/N (see (20)) is not sufficient for $|f(Z) - f(Z')|$. For any constant $C_1 > 0$ there exists $\tilde{N} = 2^{C_1}$, such that for every $N > \tilde{N}$ the following inequality is satisfied:

$$\frac{C_1}{N} < \frac{log_2 N}{N} \le |f(Z) - f(Z')|. \tag{28}$$

3 McDiarmid's Bound for Information Gain

In this section, we assume that the split measure is information gain. The following theorem guarantees that a decision tree learning system, applied to data streams, has the property that its output is nearly identical to that produced by a conventional learner.

Theorem 1 : Let $Z = \{X_1, \cdots, X_N\}$ be the set of independent random variables, with each of them taking values in the set $A \times B \times \cdots \times \Lambda$. Then, for any fixed δ and any pair of attributes a and b, where $f(Z) = Gain_a(Z) - Gain_b(Z) > 0$, if

$$\epsilon = C_{Gain}(K, N)\sqrt{\frac{\ln(1/\delta)}{2N}}, \tag{29}$$

then

$$Pr(f(Z) - E[f(Z)] > \epsilon) \le \delta, \tag{30}$$

where

$$C_{Gain}(K, N) = 6(K\log_2 eN + \log_2 2N) + 2\log_2 K.$$

Proof. see Appendix I, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.66. □

Corollary 1. *Suppose a and b are attributes for which the values of information gain, calculated from the data sample Z, satisfy $Gain_a(Z) > Gain_b(Z)$. For any fixed δ and ϵ given by (29), if $f(Z) > \epsilon$, then with probability $1 - \delta$ attribute a is better to split than attribute b, according to whole data stream. Moreover, if a and b are attributes with the highest and the second highest values of information gain, then with probability $1 - \delta$, a is the best attribute to split according to the whole stream.*

Proof. For this particular choice of attributes a and b the assumptions of Theorem 1 are satisfied. Therefore, inequality (30) holds and can be transformed to the form

$$P(E[f(Z)] \ge f(Z) - \epsilon) \ge 1 - \delta. \tag{31}$$

Notice that if $f(Z) > \epsilon$, then $E[f(Z)] > 0$ with probability $1 - \delta$. The inequality $E[f(Z)] > 0$ is equivalent to $E[Gain_a(Z)] > E[Gain_b(Z)]$ what means that the attribute a is better than attribute b to split, according to the whole data stream, with probability $1 - \delta$.

Let us consider now the case, where a and b are attributes with the highest and the second highest values of information gain, respectively. Then for any other attribute c ($c \neq a$), by virtue of Theorem 1, $E[Gain_a(Z)] > E[Gain_c(Z)]$ with probability $1 - \delta$. □

Example 1. We want to check if attribute a is better to split than attribute b, with probability $1 - \delta = 0.95$, using information gain given by (8). Assume that we have set Z of 500,000 data and each of them is assigned to one of 3 classes.

Now we have to compute ϵ using (29). It is easy to check that for our data ϵ is equal to 0.847229. Since $K = 3$, $Gain_a(\cdot) - Gain_b(\cdot) \in [0, \log_2 3]$. In our case, if $Gain_a(Z) - Gain_b(Z) \in (0.847229; \log_2 3]$, then attribute a is better to split the node than attribute b, with probability 0.95. Otherwise, for $Gain_a(Z) - Gain_b(Z) \in [0; 0.847229]$, we have $f(Z) \le \epsilon$. Therefore, we cannot use inequality (31) to determine which attribute is better to split.

Example 2. Let us assume that the calculated value of $f(Z) = Gain_a(Z) - Gain_b(Z)$ equals 2.28885 and the number of classes $K = 8$. We want to know if our amount of data N is sufficient to say that attribute a is better to split than attribute b, with probability $1 - \delta = 0.95$. According to inequality (31), N is a sufficient number if $f(Z) > \epsilon$, where ϵ is given by (29). Formula (29) does not allow to determine N analytically, however the problem can be solved numerically. In our case, for $N = 327{,}741$ the value of ϵ is 2.288853 and it is greater than $f(Z)$. It means that N is not large enough. However, for $N = 327,742$ we obtain $\epsilon = 2.2888496$ and it satisfies the condition $f(Z) > \epsilon$. In this case N is large enough to say that attribute a is better to split than attribute b, with probability 0.95.

4 McDiarmid's Bound for Gini Index

Gini index for a set Z is defined as follows:

$$Gini(Z) = 1 - \sum_{j=1}^{K}\left(\frac{n^j}{N}\right)^2. \tag{32}$$

Let $A = \{a_1, \dots, a_{|a|}\}$ be a set of values for attribute a. There are $2^{|a|-1} - 1$ different ways of partitioning set A into two disjoint subsets: $A_1 = \{a_{j_i} \in A : i \in \{1, \dots, D\}\}$ and $A_2 = \{a_{j_i} \in A : i \in \{D+1, \dots, |a|\}\} = A \setminus A_1$. Since $A = A_1 \cup A_2$, it is sufficient to consider only one of these sets, e.g., A_1. The set Z (Z') is partitioned into two disjoint subsets Z_1 and Z_2 (Z'_1 and Z'_2), according to sets A_1 and A_2. Let us define

$$\begin{aligned} Gini_{A_1}(Z) = \frac{n_1}{N}\left(1 - \sum_{j=1}^{K}\left(\frac{n_1^j}{n_1}\right)^2\right) \\ + \frac{n_2}{N}\left(1 - \sum_{j=1}^{K}\left(\frac{n_2^j}{n_2}\right)^2\right) = Gini_{A_2}(Z), \end{aligned} \tag{33}$$

where $n_l, l = 1, 2$, is a number of elements in set Z_l and n_l^j is a number of elements in set Z_l from class k_j.

Among all the possible partitions of set A into two disjoint sets A_1 and A_2, we choose one with the lowest value of $Gini_{A_1}(Z)$. This value is a Gini index for attribute a

$$Gini_a(Z) = \min_{A_1}\{Gini_{A_1}(Z)\}. \tag{34}$$

We will call it an optimal partition.

A split measure for attribute a, based on Gini index (Gini gain), is given by

$$\Delta Gini_a(Z) = Gini(Z) - Gini_a(Z). \tag{35}$$

Let us assume that a is an attribute with the highest value of $\Delta Gini_a(Z)$ and b is the second-best attribute. Define

$$\begin{aligned} f(Z) &= \Delta Gini_a(Z) - \Delta Gini_b(Z) \\ &= Gini_b(Z) - Gini_a(Z). \end{aligned} \tag{36}$$

The interpretation of the following theorem is the same as that of Theorem 1.

Theorem 2 : Let $Z = \{X_1, \cdots, X_N\}$ be the set of independent random variables, with each of them taking values in the set $A \times B \times \cdots \times \Lambda$. Then, for any fixed δ and any pair of attributes a and b, where $f(Z) = Gini_b(Z) - Gini_a(Z) > 0$, if

$$\epsilon = 8\sqrt{\frac{\ln(1/\delta)}{2N}} \tag{37}$$

then

$$Pr(f(Z) - E[f(Z)] > \epsilon) \leq \delta \tag{38}$$

Proof. Appendix II, which is available in the online supplemental material. □

Corollary 2. *Suppose a and b are attributes for which the values of Gini gain, calculated from the data sample Z, satisfy $\Delta Gini_a(Z) > \Delta Gini_b(Z)$. For any fixed δ and ϵ given by (37), if $f(Z) > \epsilon$, then with probability $1 - \delta$ attribute a is better to split than attribute b, according to whole data stream. Moreover, if a and b are attributes with the highest and the second highest values of Gini gain, then with probability $1 - \delta$, a is the best attribute to split according to the whole stream.*

Proof is analogous to the proof of Corollary 1.

Example 3. Let us assume that we have set Z of 10,000 data. We want to check if attribute a is better to split than attribute b, with probability $1 - \delta = 0.95$. To check this, we use expressions (37) and (38).

First let us compute the value of ϵ given by (37). For our data $\epsilon = 0.0979099$. Note that in this case, contrary to Theorem 1, the value of ϵ does not depend on the number of classes K. According to Corollary 2 attribute a is better to split the node than attribute b, with probability $1 - \delta$, if $f(Z) > \epsilon$. In our case this condition is satisfied for $Gini_b(Z) - Gini_a(Z) > 0.0979099$.

If $Gini_b(Z) - Gini_a(Z) \leq \epsilon$, then we cannot determine which attribute is better.

Example 4. Let us assume that the calculated value of $f(Z) = Gini_b(Z) - Gini_a(Z)$ is equal to 0.354. We want to know how many data elements N we should have, to say that attribute a is better to split than attribute b with probability $1 - \delta = 0.975$. According to inequality (31), with ϵ given by (37), N is a sufficient number if

$$f(Z) > 8 \cdot \sqrt{\frac{\ln(1/\delta)}{2N}}. \tag{39}$$

Contrary to the case of information gain (see Example 2), N can be determined using simple algebra

$$N > \frac{64 \ln(1/\delta)}{2 \cdot (f(Z))^2}. \tag{40}$$

In our case the number of elements N should satisfy $N > 941.9718$. So if we have 942 elements or more, we can say that with probability 0.975 attribute a is better to split than attribute b.

5 The McDiarmid Tree Algorithm

In [6], Domingos and Hulten developed two algorithms for stream data mining: the Hoeffding Tree algorithm and the VFDT algorithm. In both cases they applied the Hoeffding's bound, with ϵ given by (1), for choosing the best attribute to split a node. Their methodology to deal with data streams is not contested in our paper, however a choice of the best attribute to split a node, as we showed in Section 1.3, cannot be based on the Hoeffding's inequality. In Table 1, we quote the pseudocode of "the Hoeffding Tree algorithm," as it was called in [6], replacing the Hoeffding's bound by the McDiarmid's bound (with ϵ given by (29) and (37) for information gain and Gini gain, respectively).

For convenience the following notations will be introduced:

- $\mathcal{A}$—set of all attributes.
- a—any attribute from set $\mathcal{A}$.
- a_{MAX1}—attribute with the highest value of $G(\cdot)$.
- a_{MAX2}—attribute with the second highest value of $G(\cdot)$.

6 Experimental Results

To evaluate the performance of the McDiarmid Tree algorithm, several simulations were conducted. Since ϵ for the Gini gain tends to zero much faster than for the

TABLE 1
The McDiarmid Tree Algorithm

```
Inputs:  $Z$        is a sequence of examples,
         $\mathcal{A}$        is a set of discrete attributes,
         $G(\cdot)$     is a split evaluation function,
         $\delta$        is one minus the desired probability of
                  choosing the correct attribute at any
                  given node.
Output:  $McDT$ is a decision tree.

Procedure McDiarmidTree($Z, \mathcal{A}, G, \delta$)
Let $McDT$ be a tree with a single leaf $l_1$ (the root). Let
$\mathcal{A}_{l_1} = \mathcal{A}$
For each class $k_r$
    For each attribute $a \in \mathcal{A}$
        For each value $a_\lambda$ of attribute $a$
            Let $n^r_{a_\lambda}(l_1) = 0$.
For each example $X$ in $Z$
    Sort $X$ into a leaf $l$ using $McDT$.
    For each attribute $a \in \mathcal{A}_l$
        For each value $a_\lambda$ of attribute $a$
            If value of example $X$ for attribute $a$
            equals $a_\lambda$ and $X$ comes from class $r$
                Increment $n^r_{a_\lambda}(l)$.
    Label $l$ with the majority class among the
    examples seen so far at $l$.
    If the examples seen so far at $l$ are not of
    the same class, then
        Compute $\overline{G}_l(a)$ for each attribute
        $a \in \mathcal{A}_l$ using the counts $n^r_{a_\lambda}(l)$.
        Let $a_{MAX1}$ be the attribute with the
        highest $\overline{G}_l$.
        Let $a_{MAX2}$ be the attribute with the
        second-highest $\overline{G}_l$.
        Compute $\epsilon$ using equation (29) for
        information gain or (37) for Gini gain.
        If $\overline{G}_l(a_{MAX1}) - \overline{G}_l(a_{MAX2}) > \epsilon$, then
            Replace $l$ by an internal node that
            splits on $a_{MAX1}$.
            For each branch of the split
                Add a new leaf $l_m$, and let
                $\mathcal{A}_{l_m} = \mathcal{A}_l \backslash \{a_{MAX1}\}$ at $l_m$.
                For each class $k_r$
                    For each attribute $a \in \mathcal{A}$
                        For each value $a_\lambda$ of $a$
                            Let $n^r_{a_\lambda}(l_m) = 0$.
Return $McDT$.
```

information gain, only the Gini gain is considered in the following experiments. Synthetic data were used, generated on a basis of synthetic decision trees. These synthetic trees were constructed in the same way as described in [6]. At each level of the tree, after the first L_{min} levels, each node is replaced by a leaf with probability ω. To the rest of nodes a splitting attribute is randomly assigned; it has to be an attribute which has not already occurred in the path from the root to the considered node. The maximum depth of the synthetic tree is L_{max} (at this level all nodes are replaced by leaves). After the whole tree is constructed, to each leaf a class is randomly assigned. Each synthetic tree represents a different data concept. Data concept is a particular distribution of attributes values and classes. In this work 12 synthetic trees were generated (all of them with $\omega = 0.15$, $L_{min} = 3$ and $L_{max} = 18$) giving twelve different data concepts. In the following simulations, for any set of McDiarmid Tree parameters (δ, N), algorithm was run 12 times, once for each synthetic data concept. Then, the final result was obtained as the average over all runs.

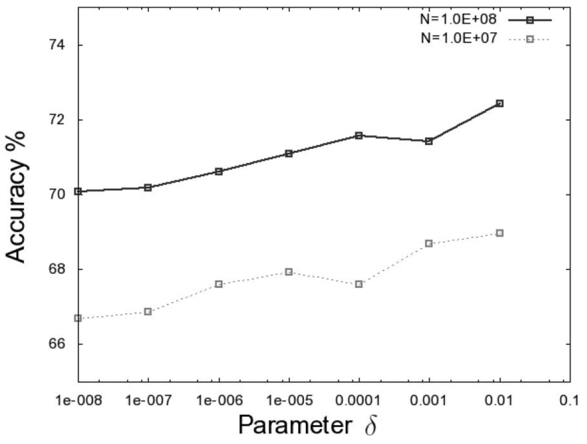

Fig. 1. Accuracy as a function of parameter δ.

First, we compare the performance of the McDiarmid Tree algorithm according to different values of parameter δ. As we can see in Fig. 1, the accuracy increases with increasing values of δ. Analyzing this relationship, we noticed that with decreasing values of δ trees become less complex. It happens because the splitting condition is more difficult to achieve. Therefore, a new mechanism should be added, that allows to split a node when there are two good attributes. We notice also, that there are only slight changes of accuracy according to δ (less than 2.5 percent).

In the next simulations, the performance of the McDiarmid Tree algorithm was compared with the CART algorithm [5], developed for static data. As it is depicted in Fig. 1, the accuracy of the McDiarmid Tree algorithm is not satisfactory. This result is mainly due to the small depth of the obtained trees. For small sizes of training data sets N, the tree remains undivided (consisting of only one node—the root). If there are two attributes with comparable, very high values of the Gini gain function, the algorithm cannot decide which one is better even for very large N. Such a situation can dramatically slow down the tree construction process. In view of this problem, we modified the McDiarmid Tree algorithm introducing the tie breaking parameter τ, following the idea of the VFDT system [6]. If $\epsilon < \tau$, the split is made. As in [6], τ was set to 0.05. Previous results (see Fig. 1) show, that the accuracy of the tree depends on δ only slightly. Thus, parameter δ can be chosen arbitrarily and we set $\delta = 10^{-7}$, as in [6]. Simulations for the McDiarmid Tree algorithm were performed for different training data set sizes, from $N = 10^4$ to $N = 10^9$. Simulations for the CART algorithm were performed maximally for $N = 10^6$, because of the random access memory limits. Results are presented in Fig. 2.

For the same value of N, the accuracy of the CART algorithm is significantly better than for the McDiarmid Tree algorithm. The accuracy increases with the growth of the size N of the training data set. For N around 10^9 high accuracy can be obtained.

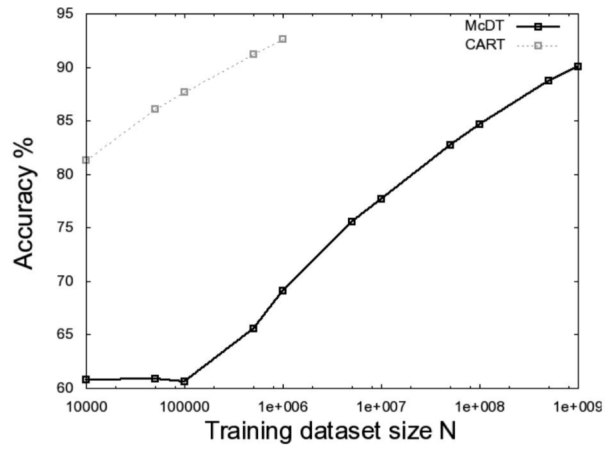

Fig. 2. Accuracy as a function of the training data set size N.

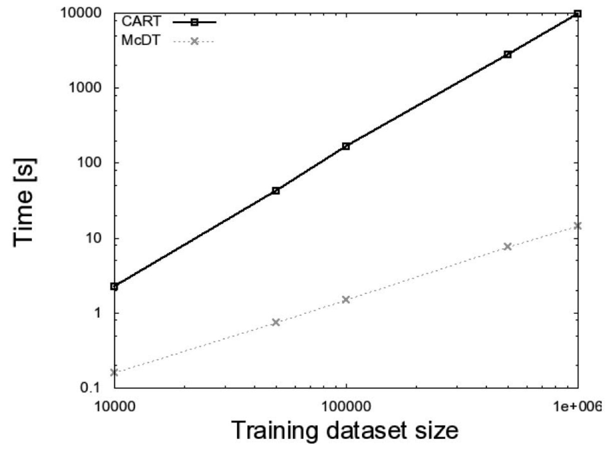

Fig. 3. Processing time as a function of training data set size N.

In our experiments also the relation between the processing time and the size of training data set was analyzed. The results are presented in Fig. 3.

As we can see, for the same number of data elements N, the CART algorithm is much more time consuming than the McDiarmid Tree algorithm. The processing time for the CART algorithm is a power function of N (note that the scale in Fig. 3 is logarithmic). For the McDiarmid Tree algorithm the relation is almost linear. Low memory consumption is the another advantage of the McDiarmid Trees. The amount of memory does not depend on the size of a training data set. For the CART algorithm the consumed memory depends linearly on N.

The above properties of the proposed method, i.e., fast processing, low memory consumption and quite good accuracy, make McDiarmid Trees a proper tool for data stream mining.

7 Final Remarks

In the paper, we showed that result (1), the most cited in data stream mining, is not valid in a general case. The appropriate tool to solve the problem is the McDiarmid's inequality and its application led to bounds given in Theorems 1 and 2. Therefore, we would suggest to use the term "McDiarmid Trees" instead of "Hoeffding Trees" in all algorithms previously developed and based on the Hoeffding's inequality. A challenge for future work is to improve bounds obtained in Theorem 1 and Theorem 2.

Acknowledgments

The authors would like to thank the reviewers for helpful comments. This paper was prepared under project operated within the Foundation for Polish Science Team Programme co-financed by the EU European Regional Development Fund, Operational Program Innovative Economy 2007-2013, and also supported by National Science Center NCN.

References

[1] C. Aggarwal, *Data Streams. Models and Algorithms.* Springer, 2007.
[2] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams.* IOS Press, 2010.
[3] A. Bifet, G. Holmes, G. Pfahringer, R. Kirkby, and R. Gavalda, "New Ensemble Methods for Evolving Data Streams," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '09),* June/July 2009.
[4] A. Bifet and R. Kirkby, *Data Stream Mining a Practical Approach,* technical report, Univ. of Waikato, 2009.
[5] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* Chapman and Hall, 1993.
[6] P. Domingos and G. Hulten, "Mining high-speed Data Streams," *Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 71-80, 2000.
[7] W. Fan, Y. Huang, and P.S. Yu, "Decision Tree Evolution using Limited Number of Labeled Data Items from Drifting Data Streams," *Proc. IEEE Fourth Int'l Conf. Data Mining,* pp. 379-382, 2004.
[8] M.M Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining Data Streams: A Review," *ACM SIGMOD Record,* vol. 34, no. 2, pp. 18-26, June 2005.
[9] J. Gama, R. Fernandes, and R. Rocha, "Decision Trees for Mining Data Streams," *Intelligent Data Analysis,* vol. 10, no. 1, pp. 23-45, Mar. 2006.
[10] J. Han and M. Kamber, *Data Mining: Concepts and Techniques,* second ed. Elsevier, 2006.
[11] W. Hoeffding, "Probability Inequalities for Sums of Bounded Random Variables," *J. Am. Statistical Assoc.,* vol. 58, no. 301, pp. 13-30, Mar. 1963.
[12] G. Hulten, L. Spencer, and P. Domingos, "Mining Time-Changing Data Streams," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 97-106, 2001.
[13] R. Kirkby, "Improving Hoeffding Trees," PhD dissertation, Univ. of Waikato, Hamilton, 2007.
[14] J.Z. Kolter and M.A. Maloof, "Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift," *Proc. IEEE Third Int'l Conf. Data Mining,* pp. 123-130, 2003.
[15] D.T. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining.* Wiley & Sons, Inc, 2005.
[16] X. Li, J.M. Barajas, and Y. Ding, "Collaborative Filtering On Streaming Data With Interest-Drifting," *Int'l Intelligent Data Analysis,* vol. 11, no. 1, pp. 75-87, 2007.
[17] J. Liu, X. Li, and W. Hong, "Ambiguous Decision Trees for Mining Concept-Drifting Data Streams," *Pattern Recognition Letters,* vol. 30, no. 15, pp. 1347-1355, Nov. 2009.
[18] C. McDiarmid, *On the Method of Bounded Differences, Surveys in Combinatorics,* J. Siemons, ed., pp. 148-188, Cambridge Univ. Press, 1989.
[19] B. Pfahringer, G. Holmes, and R. Kirkby, "New Options for Hoeffding Trees," *Proc. 20th Australian Joint Conf. Advances in Artificial Intelligence,* pp. 90-99, 2007.
[20] J.R. Quinlan, "Learning Efficient Classification Procedures and Their Application to Chess End Games," *Machine Learning: An Artificial Intelligence Approach,* vol. 1, pp. 463-482, 1983.
[21] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[22] L. Rutkowski, *Computational Intelligence: Methods and Techniques.* Springer-Verlag, 2008.
[23] L. Rutkowski, *New Soft Computing Techniques for System Modelling.* Springer-Verlag, 2004.

[24] L. Rutkowski, "Adaptive Probabilistic Neural-Networks for Pattern Classification in Time-Varying Environment," *IEEE Trans. Neural Networks,* vol. 15, no. 4, pp. 811-827, July 2004.

[25] L. Rutkowski, "Generalized Regression Neural Networks in Time-Varying Environment," *IEEE Trans. Neural Networks,* vol. 15, no. 3, pp. 576-596, May 2004.

[26] A. Tsymbal, "The Problem of Concept Drift: Definitions and Related Work," Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Apr. 2004.

[27] H. Wang, W. Fan, P.S. Yu, and J. Han, "Mining Concept-Drifting Data Streams Using Ensemble Classifiers," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 226-235, 2003.

Leszek Rutkowski (F'05) received the MSc and PhD degrees both from the Technical University of Wroclaw, Poland, in 1977 and 1980, respectively. Since 1980, he has been with the Technical University of Czestochowa, where he is currently a professor and chairman of the Computer Engineering Department. From 1987 to 1990, he held a visiting position in the School of Electrical and Computer Engineering at Oklahoma State University. His research interests include neural networks, fuzzy systems, computational intelligence, pattern classification, and expert systems. In May and July 2004, he presented in the *IEEE Transaction on Neural Networks* a new class of probabilistic neural networks and generalized regression neural networks that worked in a time-varying environment. He has published more than 170 technical papers including 20 in various IEEE Transactions. He is the author of the following books: *Computational Intelligence* published by Springer (2008), *New Soft Computing Techniques For System Modeling, Pattern Classification and Image Processing* published by Springer (2004), *Flexible Neuro-Fuzzy Systems* published by Kluwer Academic Publishers (2004), *Methods and Techniques of Artificial Intelligence* (2005, in Polish), *Adaptive Filters and Adaptive Signal Processing* (1994, in Polish), and coauthor of two others (1997 and 2000, in Polish) *Neural Networks, Genetic Algorithms and Fuzzy Systems* and *Neural Networks for Image Compression*. He is a president and founder of the Polish Neural Networks Society. He organized and served as a general chair of the International Conferences on Artificial Intelligence and Soft Computing held in 1996, 1997, 1999, 2000, 2002, 2004, 2006, 2008, and 2010. He is past associate editor of the *IEEE Transactions on Neural Networks* (1998-2005) and *IEEE Systems Journal* (2007-2010). He is editor-in-chief of the *Journal of Artificial Intelligence and Soft Computing Research* and he is on the editorial board of the *International Journal of Applied Mathematics and Computer Science* (1996-present) and the *International Journal of Biometric* (2008-present). In 2004, he was awarded by the IEEE Fellow Membership Grade for contributions to neurocomputing and flexible fuzzy systems. He is a recipient of the *IEEE Transactions on Neural Networks* 2005 Outstanding Paper Award. He served in the IEEE Computational Intelligence Society as the chair of the Distinguished Lecturer Program (2008-2009) and the chair of the Standards Committee (2006-2007). He is the founding chair of the Polish Chapter of the IEEE Computational Intelligence Society which won 2008 Outstanding Chapter Award. In 2004, he was elected as a member of the Polish Academy of Sciences.

Lena Pietruczuk received the MS degree in mathematics (genetic algorithms and their use in forecasting demand) from the Department of Mathematics and Computer Science, University of Wroclaw, in 2010 and currently she is working toward the PhD degree in computer science in Department of Computer Engineering at Czestochowa University of Technology, Poland. Her current research interests include data stream mining, neural networks, and evolutionary algorithms.

Piotr Duda received the MSc degree from the Department of Mathematics, Physics and Chemistry, University of Silesia in Katowice, Poland, in 2009. Currently, he is working toward the PhD degree in computer science in the Department of Computer Engineering at Czestochowa University of Technology, Poland. His current research interests include statistics and data stream mining, especially classification problems.

Maciej Jaworski received the MSc (Hons) degree in theoretical physics from the Jagiellonian University, Krakow, in 2009, and the MSc degree in applied computer science from the University of Science and Technology, Krakow, in 2011. Currently, he is working toward the PhD degree in computer science in the Department of Computer Engineering at Czestochowa University of Technology, Poland. His current research interests include computational intelligence and data stream mining.